UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/729,666	12/05/2003	Jason D. Sollom	RA 5512(1056-004US01)	3160

7590 10/23/2006

Charles A. Johnson
Unisys Corporation
P O Box 64942 MS 4773
St. Paul, MN 55164

EXAMINER	
GEBRESILASSIE, KIBROM K	

ART UNIT	PAPER NUMBER
2128	

DATE MAILED: 10/23/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/729,666	SOLLOM ET AL.	
	Examiner	Art Unit	
	Kibrom K. Gebresilassie	2128	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 August 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is responsive to amended application filed on August 2, 2006.

2. Claims 1, 14, and 27 are amended.

3. Claims 1-30 are examined.

Response to Arguments

4. Applicant's arguments filed August 2, 2006 have been fully considered.

Regarding Applicants response to 103 rejection: Applicant's arguments with respect to claims 1-30 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 13, and 26 recite the limitation "the addresses". There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 1-30 are rejected under 35 U.S.C. 102(e) as being anticipated by US 2004/0230949 A1 issued to Talwar et al, herein referred as **Talwar.**

As per Claim 1 (Currently Amended):

Talwar discloses a processor-based method performed by software emulating an instruction processor (analogous to "computer system can be utilized to implement emulation verification architecture and Java verification system" [0045] lines 1-6; Fig. 3A), the method comprising:

processing read instructions with an emulated processor executing within an emulation environment to output independent read requests via an operand interface and an op-code interface of the emulated processor to independently fetch op-codes and operands (analogous to "execution engine 234 executes bytecode instructions included in emulation class information 220. In one exemplary implementation, each **bytecode instruction includes an op-code** that indicates the operation to be performed and can **include an operand** that provides information associated with the operation. Execution engine **234 fetches an op-code and related information** (e.g. **operand and/or data stored in other area**) and performed the instruction indicated by the op-code." [0042]; See Fig. 1C element 132A and 132B with different interfaces from two different shared libraries) from an emulated memory external from the emulated processor (analogous to "**communication bus 357** is **coupled** to **central processor 351, memory 352**,..." [0045] and "**Memory 352** stores information and instruction, **including instructions for validating the legitimacy of native language code**." [0045]; See Fig. 3A element 351 and 351);

independently comparing op-code reference data and operand reference data to operands and op-codes received in response to the read requests (analogous to "**Verifier 122** can perform predetermined algorithms on received information (e.g. **from native shared library 130 and Java class 110**) to establish generated values and then **compare the generated values to the received verification information** (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B..."); and

recording results of the independent comparisons (analogous to "**Emulation class loading module 231** verifies emulation class and native language information and **places emulation class information into memory**...."[0039]).

As per Claim 2:

Talwar discloses storing the op-code reference data and the operand reference data within a set of data memories of the emulated instruction processor (analogous to "Emulation class loading module 231 verifies emulation class and native language information and places emulation class information into memory...."[0039]);

maintaining within the emulated instruction processor an operand data pointer to address the operand reference data and an op-code pointer to address the op-code reference data (analogous to "...the components of emulation verification system cooperatively operate to verify and load native language information. Verifier determines if the verification information indicates native language code requested by emulation class is legitimate by verifying information included in native language shared library conforms with the predetermined relationship." [0027]); and

independently accessing the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands (analogous to "execution engine 234 executes bytecode instructions included in emulation class information 220. In one exemplary implementation, each bytecode instruction includes an op-code that indicates the operation to be performed and can include an operand that provides information associated with the operation. Execution engine 234 fetches an op-code and related information (e.g. operand and/or data stored in other area) and performed the instruction indicated by the op-code." [0042]; See Fig. 1C element 132A and 132B with different interfaces from two different shared libraries).

As per Claim 3:

Talwar discloses when processing the read instructions, determining whether each of the read instructions required an operand read request or an op-code read request; and independently updating the op-code reference pointer or the operand reference pointer based on the determination (analogous to "...the examination confirms if native language code is uncorrupted (e.g. ...pointers to incorrect memory locations,..)...Permission to load the requested native language code is granted..." [0057]).

As per Claim 4:

Talwar discloses the read instructions form part of an instruction stream executed by the emulated instruction processor, the method further comprising: processing a flow control instruction of the instruction stream with the emulated instruction processor; and

upon processing the flow control instruction, synchronizing the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code reference data and a portion of the operand reference data associated with a target address of the flow control instruction (analogous to "During bytecode verification the Java virtual machine can perform a data flow analysis on bytecode streams that represent methods of class." [0066]).

As per Claim 5:

Talwar discloses, compiling test software to output the operand reference data, the op-code reference data, and the instruction stream (analogous to "compiler, compiles the bytecode information..."[0042]).

As per Claim 6:

Talwar discloses independently comparing further comprises: latching the op-code reference data within a first latch within the emulated instruction processor; and comparing the latched op-code referenced data and the received op-codes with a comparator to produce the results (analogous to "Verifier 122 can perform predetermined algorithms on received information (e.g. from native shared library 130 and Java class 110) to establish generated values and then compare the generated values to the received verification information (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 7:

Talwar discloses latching the operand reference data within a first latch within the emulated instruction processor; and comparing the latched operand referenced data and the received operand with a comparator to produce the results requests (analogous to "Verifier 122 can perform predetermined algorithms on received information (e.g. from native shared library 130 and Java class 110) to establish generated values and then compare the generated values to the received verification information (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 8:

Talwar discloses storing write data within a data memory of the emulated instruction processor; maintaining within the emulated instruction processor a write data pointer to address the write data; and processing a write instruction with the emulated processor to output a write request via a data interface of the emulated processor, wherein the write request comprises a portion of the write data referenced by the write pointer (analogous to "...the examination confirms if native language code is uncorrupted (e.g. ...pointers to incorrect memory locations,..)...Permission to load the requested native language code is granted..." [0057]).

As per Claim 9:

Talwar discloses generating a report based on the results, wherein the report identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data (analogous to "Verified 22 issues load denial indicator 25 to prevent loading of native

language code. If the verification information does exhibit the predetermined correspondence, verifier 22 issues load permission indicator 27 to continue with loading of native language code." [0027]; Fig. 1A elements 25 and 27).

As per Claim 10:

Talwar discloses storing addresses associated with received operands and the op-codes, copies of the received operands or op-codes, copies of the reference operands and the reference op-codes, or copies of the instruction (analogous to "Runtime data area 233 defines and tracks assignment of logical memory locations associated with executing method and processing data of loaded classes."[0038]).

As per Claim 11:

Talwar discloses recording the results comprises recording the result within a register within the emulated instruction processor (analogous to "Emulation class loading module 231 verifies emulation class and native language information and places emulation class information into memory...."[0039]; Fig. 2A element 233).

As per Claim 12:

Talwar discloses applying bit masks to at least a portion of a result of the independent comparisons of the op-code reference data and the operand reference data to the op-code and the operand received in response to the read requests (analogous to "The fields or class variables ("variable") can include data (e.g., integers r characters) which can be private data accessible..." [0030]).

As per Claim 13:

Talwar discloses storing the addresses of a small number of received operands and received op-codes within the emulated instruction processor (analogous to "Emulation class loading module 231 verifies emulation class and native language information and places emulation class information into memory...."[0039]); and selectively enabling and disabling access to the cache when executing subsequent read instructions and waiting to output read requests via the operand interface and the op-code interface when the read instructions request operands and op-codes are invalidated within the cache based on a configurable option (analogous to "Verified 22 issues load denial indicator 25 to prevent loading of native language code. If the verification information does exhibit the predetermined correspondence, verifier 22 issues load permission indicator 27 to continue with loading of native language code." [0027]; Fig. 1A elements 25 and 27).

As per Claim 14 (Currently Amended):

Talwar discloses a processor-based system for emulating an instruction processor (analogous to "computer system can be utilized to implement emulation verification architecture and Java verification system" [0045] lines 1-6; Fig. 3A) comprising:

a computing system to provide an emulation environment (analogous to "**Emulation verification system**" [0045]; see Fig. 3A element 350); and

software execution within the emulation environment to emulate an instruction processor having an operand interface and an op-code interface (analogous to "execution engine 234 executes bytecode instructions included in emulation class

information 220. In one exemplary implementation, **each bytecode instruction includes an op-code** that indicates the operation to be performed and can **include an operand** that provides information associated with the operation. Execution engine 234 **fetches an op-code and related information** (e.g. **operand and/or data stored in other area**) and performed the instruction indicated by the op-code." [0042]; See Fig. 1C element 132A and 132B with different interfaces from two different shared libraries) and to emulate a memory external to the instruction processor (analogous to . "**communication bus** 357 is **coupled** to **central processor** 351, **memory** 352,..." [0045] and "**Memory 352 stores** information and instruction**, including instructions for validating the legitimacy of native language code**." [0045]; See Fig. 3A element 351 and 351),

wherein the software emulates the instruction processor by processing read instructions and outputting corresponding read requests on the operand interface or the op-code interface (analogous to "execution engine 234 executes bytecode instructions included in emulation class information 220. In one exemplary implementation, each bytecode instruction includes an op-code that indicates the operation to be performed and can include an operand that provides information associated with the operation. Execution engine 234 fetches an op-code and related information (e.g. operand and/or data stored in other area) and performed the instruction indicated by the op-code." [0042]; See Fig. 1C element 132A and 132B with different interfaces from two different shared libraries) to independently fetch respective op-codes or operands from the emulated memory (analogous to "communication bus 357 is coupled to central

processor 351, memory 352,..." [0045] and "Memory 352 stores information and instruction, including instructions for validating the legitimacy of native language code." [0045]; See Fig. 3A element 351 and 351), and

independently comparing op-code reference data and operand reference data to operands and op-codes received from the operand interface and op-code interface in response to the read requests (analogous to "**Verifier 122** can perform predetermined algorithms on **received information (e.g. from native shared library 130 and Java class 110)** to establish generated values and then **compare the generated values to the received verification information** (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 15:

Talwar discloses software emulates the instruction processor by recording results of the independent comparisons within a register of the emulated instruction processor (analogous to "Emulation class loading module 231 verifies emulation class and native language information and places emulation class information into memory...."[0039]).

As per Claim 16:

Talwar discloses the emulated instruction processor comprises: a first data memory to store the op-code reference data; and a second data memory to store the operand reference data (analogous to "Native shared library 130A and native shared Library 130B" See Fig. 1C).

As per Claim 17:

Talwar discloses the emulated instruction processor comprises: a control unit that maintains an operand data pointer to address the operand reference data within the first data memory and an op-code pointer to address the op-code reference data within the second data memory, wherein the control unit independently accesses the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands (analogous to "execution engine 234 executes bytecode instructions included in emulation class information 220. In one exemplary implementation, each bytecode instruction includes an op-code that indicates the operation to be performed and can include an operand that provides information associated with the operation. Execution engine 234 fetches an op-code and related information (e.g. operand and/or data stored in other area) and performed the instruction indicated by the op-code." [0042]; See Fig. 1C element 132A and 132B with different interfaces from two different shared libraries).

As per Claim 18:

Talwar discloses upon processing the read instructions, the control unit determines whether each of the read instructions required an operand read request or an op-code read request, and independently updates the op-code reference pointer or the operand reference pointer based on the determination (analogous to "native code usually has pointers and the pointers can be altered..."[0007]).

As per Claim 19:

Talwar discloses the read instructions form part of an instruction stream executed by the emulated instruction processor, and upon processing a flow control instruction of the instruction stream, the control unit synchronizes the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code reference data and a portion of the operand reference data associated with a target address of the flow control instruction (analogous to "During bytecode verification the Java virtual machine can perform a data flow analysis on bytecode streams that represent methods of class." [0066]).

As per Claim 20:

Talwar discloses a compiler executing on the computing system to compile test software to output the operand reference data, the op-code reference data, and the instruction stream for execution by the emulated instruction processor (analogous to "compiler, compiles the bytecode information..."[0042]).

As per Claim 21:

Talwar discloses the emulated instruction processor further comprises: a latch to latch the op-code reference data from the first data memory; and a comparator to compare the latched op-code referenced data and the received op-codes (analogous to "Verifier 122 can perform predetermined algorithms on received information (e.g. from native shared library 130 and Java class 110) to establish generated values and then compare the generated values to the received verification information (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 22:

Talwar discloses the emulated instruction processor further comprises: a latch to latch the operand reference data from the second data memory; and a comparator to compare the latched operand referenced data and the received operands (analogous to "Verifier 122 can perform predetermined algorithms on received information (e.g. from native shared library 130 and Java class 110) to establish generated values and then compare the generated values to the received verification information (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 23:

Talwar discloses the emulated instruction processor further comprises: a data memory to store write data; and a control unit to maintain a write data pointer to address the write data wherein the control unit processes a write instruction to output a write request via a data interface of the emulated processor, and further wherein the control unit generates the write request to comprises a portion of the write data referenced by the write pointer (analogous to "...the components of emulation verification system cooperatively operate to verify and load native language information. Verifier determines if the verification information indicates native language code requested by emulation class is legitimate by verifying information included in native language shared library conforms with the predetermined relationship." [0027]).

As per Claim 24:

Talwar discloses emulation control software executing on the computing system to generate a report that presents/identifies any of the received op-code that do not match the op-code reference data and any of the received operands that do not match the operand reference data (analogous to "Verified 22 issues load denial indicator 25 to prevent loading of native language code. If the verification information does exhibit the predetermined correspondence, verifier 22 issues load permission indicator 27 to continue with loading of native language code." [0027]; Fig. 1A elements 25 and 27).

As per Claim 25:

Talwar discloses a set of memories to store bit masks; and bit masks to compare results of the comparison of the received operands and the received op-codes to the reference operands and the reference op-codes to mask portions of these comparisons ([0007] lines 30-33 and (analogous to "The fields or class variables ("variable") can include data (e.g., integers r characters) which can be private data accessible..." [0030]).

As per Claim 26:

Talwar discloses a cache to store the addresses of the received operands and received op-codes, wherein the emulated instruction processor selectively waits to issue read requests for subsequent read instructions via the operand interface and the op-code interface until the read instructions request operands and op-codes become invalidated within this cache based on a configurable option (analogous to "Verified 22 issues load denial indicator 25 to prevent loading of native language code. If the verification information does exhibit the predetermined correspondence, verifier 22

issues load permission indicator 27 to continue with loading of native language code." [0027]; Fig. 1A elements 25 and 27).

As per Claim 27 (Currently Amended):

Talwar discloses a processor-based system for emulating an instruction processor (analogous to "computer system can be utilized to implement emulation verification architecture and Java verification system" [0045] lines 1-6; Fig. 3A) comprising:

compiling means for compiling test software to produce operand reference data, op-code reference data, and an instruction stream having read instructions (analogous to "**emulation compiler 202, and native language compiler 204**" [0043]; Fig. 2B); and

emulating means for emulating an instruction processor having an operand interface and op-code interface and for emulating a memory external to the instruction processor (analogous to "**emulation verification system 350**" Fig. 3A), wherein the emulating means comprises:

controlling means for controlling the emulated instruction processor to process the read instructions and output corresponding read requests on the operand interface or the op-code interface to independently fetch respective operands or op-codes from the emulated memory (analogous to "**Processor 351** processes information and instructions, including instructions for validating the legitimacy of native language code." [0045]; Fig. 3A element 351),

receiving means for receiving operands and op-codes from the operand interface and op-code interface in response to the read requests (analogous to

"**Verifier 122 can** perform predetermined algorithms on **received information** (e.g. from native shared library 130 and Java class 110)..." [0033]; Fig. 1C element 122), and

comparing means for independently comparing the op-code reference data and the operand reference data to the received (analogous to "**Verifier 122** can perform predetermined algorithms on received information (e.g. from native shared library 130 and Java class 110) to establish generated values and **then compare the generated values** to the received verification information (e.g. native language verification information 112 and bytecode verification information 132.)" [0033]; See, Fig. 1C element 131A, 131B, 111A, and 111B...").

As per Claim 28:

The limitation of claim 28 has already been discussed in the rejection of Claim 16. It is therefore rejected under the same rationale.

As per Claim 29:

Talwar discloses first referencing means for addressing the operand reference data within the first storing means; and second referencing means for addressing the op-code reference data within the second storing means (analogous to "Native shared library 130A and native shared Library 130B" See Fig. 1C).

As per Claim 30:

The limitation of claim 30 has already been discussed in the rejection of Claim 9. It is therefore rejected under the same rationale.

Conclusion

9. Claims 1-30 are rejected.

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

US Patent No. 6,401,155 issued to Saville et al.

US Patent No. 6,647,489 issued to Col et al.

11. Any inquiring concerning this communication or earlier communication from the examiner should be directed to Kibrom K. Gebresilassie whose telephone number is (571) 272-8571. The examiner can normally be reached on Monday-Friday, 8:30 am to 4:30 pm. If attempts to reach the examiner by telephone are unsuccessful, the examiner supervisor, Kamini Shah can be reached at (571) 272-2279. The official fax number is (571) 273-8300. Any inquiring of a general nature relating to the status of this application should be directed to the group receptionist whose telephone number is (571) 272-3700.

Kibrom K. Gebresilassie
Patent Examiner
U.S. Patent and Trademark Office
Simulation and Emulation, Art Unit 2128
401 Dulany St., Room 5C25 (Randolph)
Alexandria, VA 22314-5774
Tel: 571-272-8571
Kibrom.gebresilassie@uspto.gov

KAMINI SHAH
SUPERVISORY PATENT EXAMINER